6.437 Project Part I

Robert Arnott

(Dated: April 30, 2018)

PROBLEM 1: FRAMEWORK

(a)

$$p_{\mathbf{y}|f}(\mathbf{y}|f) = p_{\mathbf{x}}(\mathbf{x} = f^{-1}(\mathbf{y})) = p_{\mathbf{x}}(f^{-1}(y_1)) \prod_{i=1}^{n-1} p_{x|x}(f^{-1}(y_{i+1})|f^{-1}(y_i)) \tag{1}$$

(b) Using $p_{\mathbf{y}|f}(\mathbf{y}|f)$ as specified in part (a), we can find $p_{f|\mathbf{y}}(f|\mathbf{y})$:

$$\begin{aligned} p_{f|\mathbf{y}}(f|\mathbf{y}) &= \frac{p_{\mathbf{y}|f}(\mathbf{y}|f)p_f(f)}{\sum_{f\in\mathcal{F}} p_{\mathbf{y}|f}(\mathbf{y}|f)p_f(f)} \\ &= \frac{p_{\mathbf{y}|f}(\mathbf{y}|f)(\frac{1}{m!})}{\sum_{f\in\mathcal{F}} p_{\mathbf{y}|f}(\mathbf{y}|f)(\frac{1}{m!})} \\ &= \frac{p_{\mathbf{y}|f}(\mathbf{y}|f)}{\sum_{f\in\mathcal{F}} p_{\mathbf{y}|f}(\mathbf{y}|f)} \end{aligned} \tag{2}$$

Where $\mathcal{F}$ is the family of all permutations f on the alphabet $\mathcal{A}$, so that $|\mathcal{F}| = (|\mathcal{A}|)! = m!$ and where we have use that f is drawn uniformly at random from $\mathcal{F}$.

This gives that the MAP estimator $\hat{f}_{MAP} = \arg\max_{f\in\mathcal{F}} p_{f|\mathbf{y}}(f|\mathbf{y})$ is $\hat{f}_{MAP} = \arg\max_{f\in\mathcal{F}} p_{\mathbf{y}|f}(\mathbf{y}|f)$, which we note, implies $\hat{f}_{MAP} = \hat{f}_{MLE}$

(c) Direct computation of f_{MAP} is infeasible as computing the normalization factor requires calculating $p_{\mathbf{y}|f}(\mathbf{y}|f)$ $m!$ times (for each possible f). In addition, for any decently long sequence, $p_{\mathbf{y}|f}(\mathbf{y}|f)$ is essentially 0.

PROBLEM 2: MARKOV CHAIN MONTE CARLO METHOD

(a) There are $\binom{m}{2}$ ways to pick the difference, and $m!$ possible choices for f_1. Further, choosing f_1 fixes f_2, since there is only one way for the two assignments to differ (swap them). Thus, since there $(m!)^2$ total possible choices for f_1, f_2, the probability that f_1 and f_2 differ by exactly two assignments is $\frac{\binom{m}{2}}{m!}$

(b) Let the set $S(f_i) = \{f_j \mid f_j \text{ differs from } f_i \text{ by exactly 2 assignments}\}$. We can use the proposal:

$$V(f'|f) = \begin{cases} \frac{\epsilon}{\binom{m}{2}} & f' \in S(f) \\ 1-\epsilon & f' = f \\ 0 & \text{otherwise} \end{cases}$$

(i.e. with probability ϵ, V is uniform over the set of permutation functions that differ by exactly two assignments, otherwise, V picks $f' = f$). Here, $\epsilon \in (0, 1)$ is a model parameter. Note that this proposal is symmetric between f' and f.

Under this proposal, the generated Markov Chain is irreducible (we can reach any $f \in \mathcal{F}$ from some $f_0 \in \mathcal{F}$, by continuing to change two assignments at a time), and the Markov chain is aperiodic (since each state can transition back to itself, with guaranteed non-zero probability). This ensures that the Markov Chain has a stationary distribution (i.e. is homogeneous), and thus by results given in lecture notes, that stationary distribution is $p_{f|\mathbf{y}}(f|\mathbf{y})$

(c) We use the following algorithm to find a decoder:

Initialize f_0 as a random draw from $\mathcal{F}$

repeat On iteration n

Generate sample f' from $V(\cdot|f_n)$

Compute $a = \min(1, \frac{p_{\mathbf{y}|f'}(\mathbf{y}|f')}{p_{\mathbf{y}|f}(\mathbf{y}|f_n)})$

Generate sample u from $\mathcal{U}([0, 1])$ (uniform distribution)

if $u \leq a$ **then**

Accept the proposed sample, and set $f_{n+1} = f'$

else

Reject the proposed sample, and set $f_{n+1} = f_n$

end if

until Desired number of iterations or convergence

Find the generated estimate $\hat{f}_{MAP} = \arg\max_{f \in \mathcal{F}_N} p_{\mathbf{y}|f}(\mathbf{y}|f)$, where $\mathcal{F}_n$ is the set of samples generated during the iterations, potentially after discarding some set of initial samples to account for "burn-in"

PROBLEM 3: IMPLEMENTATION

(a) We obtain the behavior shown in Figure 1.

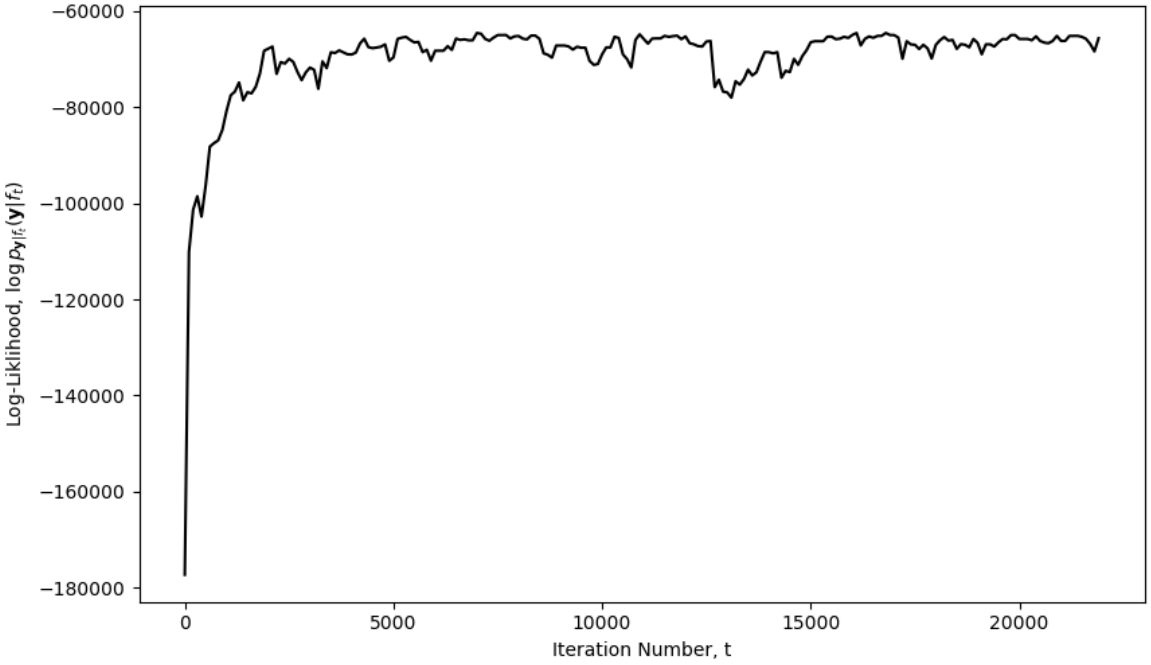

FIG. 1. The evolution of the log-likelihood $\log p_{\mathbf{y}|f}(\mathbf{y}|f_t)$ over the course of the run. Burn-in samples are included.

(b) We obtain the behavior shown in Figure 2.

(c) We obtain the behavior shown in Figure 3.

(d) We note that in general, the longer the sequence we allow our algorithm to run on, the higher the observed accuracy. This is intuitively true as we have that longer sequences will tend to behave more like the true distribution of the alphabet, and thus make it easier to separate out different permutation functions. This is shown in the empirical results shown in Figure 4.

(e) We obtain the behaviour shown in Figure 5. We note that the per character log-likelihood converges to a value slightly larger than the negative of the entropy of the letter distribution.

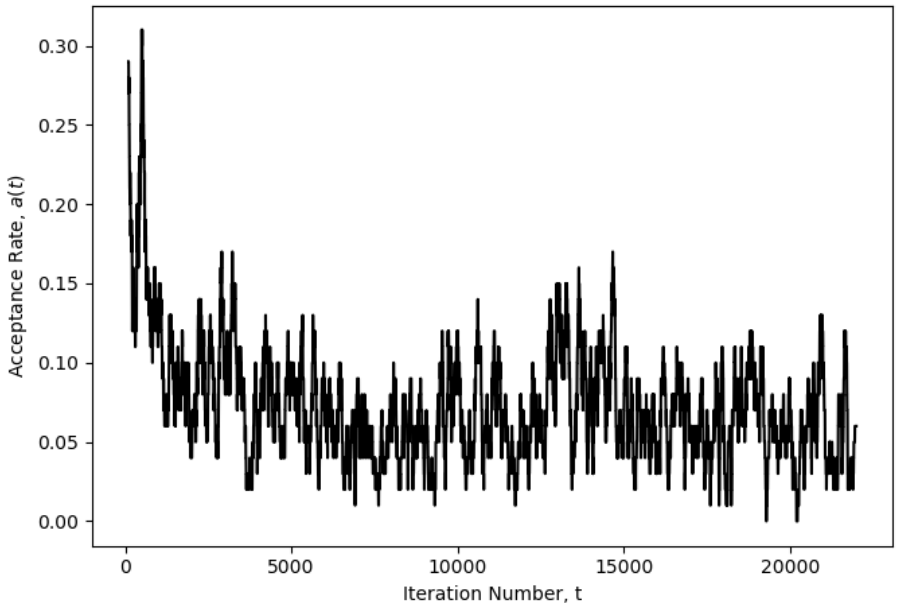

FIG. 2. The evolution of the acceptance rate $a(t)$ over the course of the run, calculated over a sliding window of 200 iterations. Burn-in samples are included.

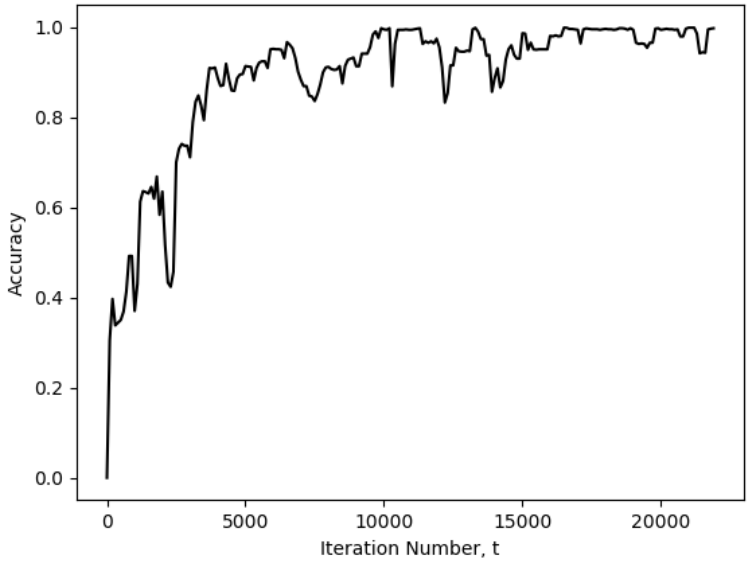

FIG. 3. The evolution of the accuracy $\beta(t)$ over the course of the run. Burn-in samples are included.

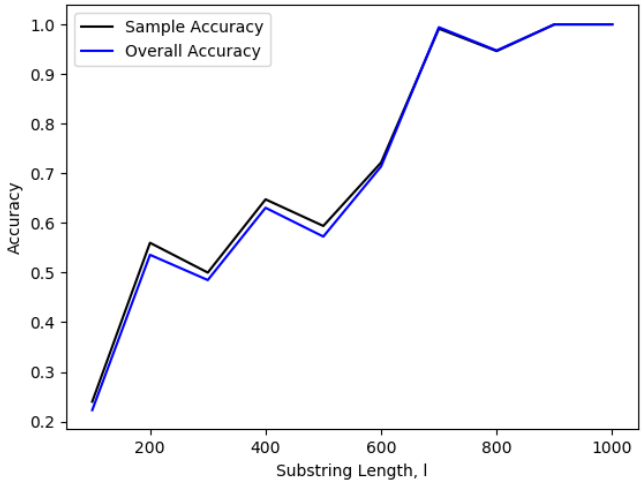

FIG. 4. The relationship between the accuracy $\beta(t)$ and the length of the sub-sequence of the ciphertext seen by the algorithm. Accuracy is measured both with respect to the sub-sequence given to the algorithm ("Sample Accuracy") and with regards to the entire ciphertext ("Overall Accuracy")

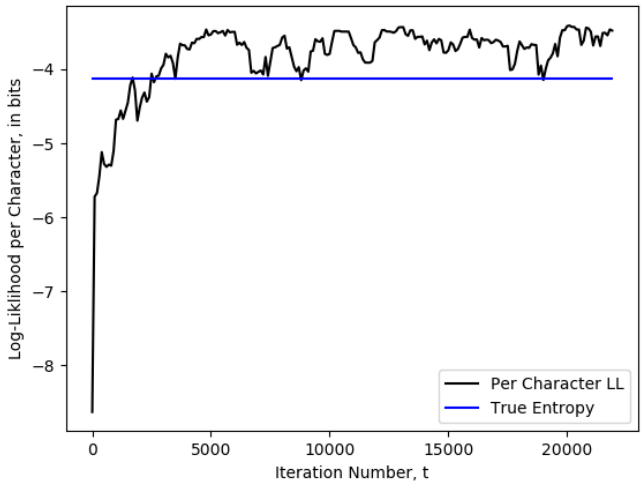

FIG. 5. The evolution of the per character log-likelihood $\frac{1}{N}\log_2 p_{\mathbf{y}|f}(\mathbf{y}|f_t)$ (measured in bits) over the course of the run, plotted alongside the negative of the entropy of the given letter distribution. Burn-in samples are included.